Các thành phần cơ bản và các kiểu dữ liệu của C

1. Danh hiệu: là tên của hằng, biến, hàm... hoặc các ký hiệu đã được quy định đặc trưng cho một thao tác nào đó. Danh hiệu có 2 loại:

- Ký hiệu: + Kí hiệu đơn: +, - , * , / ….

  + Ký hiệu kép: ++, --, ||, && …

- Danh hiệu+ Danh hiệu chuẩn: là các từ khóa mà c đã khai báo và thiết kế sẵn: main, if, while (tất cả đều là chữ thường)

  + Danh hieeuh kh chuẩn: là do ng lập trình khai báo (kh bao gồm danh hiệu chuẩn), bao gồm (chữ, số và ký tự "_"), phải bắt đầu = chữ. Trong C++ thì được tối đa 55 ký tư.

2. Các kiểu dữ liệu chuẩn của C: +char (chữ, 8 bits). Vd: char d; d = 'a'; printf("%c", d); printf("%d", d);

   + int (số nuyên, 16 bits). Vd: int a; a= 15; printf("%d", a);

   +float (thực, 32 bits). Vd: float a = 16.5; printf("%f", a);

   + double (thực, 32 bits). Vd: double a = 16.5; printf("%lf", a);

Để bổ sung cho bốn kiểu dữ liệu cơ bản C còn đưa ra các dạng bổ sung signed, unsigned, short, long :

Dạng / Kiểu	signed	unsigned	short	long
char	signed char → char	unsigned char	x	x
int	signed int → int	unsigned int → unsigned	short int → int/short	long int → long
float	x	x	x	long float → double
double	x	x	x	long double

KIỂU	KÍCH THƯỚC	TẦM TRỊ BIỂU DIỄN
unsigned char	8 bit	0..255
char	8 bit	-128..+ 127
unsigned short	16 bit	0..65535
short	16 bit	-32768..+ 32767
unsigned	16 bit	0..65535
int	16 bit	-32768..+ 32767
unsigned long	32 bit	0.. 4294967295
long	32 bit	-2147483648..+ 2147483647
float	32 bit	-3.4 E 37..3.4 E + 38
double	64 bit	-1.7 E 307..1.7 E + 308
long double	80 bit	-3.4 E 4932..1.1 E + 4932

Hằng (constant) ///

3. Biến (variable): Tất cả các biến được sử dụng trong một chương trình C đều phải được khai báo trước. Cachs khai báo: <kiểu_dữ_liêu> <tên_biến>. vd: int a; int a = 5; char b; ….
4. Các phép toán của C

- Toán tử số học: +Toán tử cộng (+) : thực hiện phép toán cộng

+Toán tử trừ (–) : thực hiện phép toán trừ

+ Toán tử nhân (*) : thực hiện phép nhân

+ Toán tử chia (/) : thực hiện phép chia

*Cộng trừ kiểu nào ra kiểu đó (nguyên * nguyên = nguyên, thực * thực = thực); nên ta phải ép kiểu vd: int a = 4, b = 3;

```
float c;
c = a / b; // c = 1
c = a*1.0/b // c= 1.333333
```

+ Toán tử modulo (%): thực hiện phép toán lấy số dư của phép chia nguyên

c = 5 % 3 // c= 2

- Toán tử quan hệ (so sánh): + Toán tử bằng (==)

+Toán tử khác (!=)

+ Toán tử lớn hơn (>)

+Toán tử nhỏ hơn (=)

+ Toán tử nhỏ hơn hoặc bằng (<=)

int a, b, c, d;

a = 1;

b = 2;

c = –3;

d = (a <= 4) * 2 + (b < 3) – c; // d = 2 + 2 – 3 = 1

- Toán tử logic: +C có các toán tử logic not (!), and (&&), or (||). Các toán tử này cho phép liên kết các biểu thức quan hệ đơn lại với nhau để tạo các biểu thức phức tạp hơn.

+ Các toán tử này có độ ưu tiên là not, and, or; nếu trong biểu thức các toán tử đều ngang cấp nhau thì thứ tự tính toán từ trái sang phải.

+ trả về true or false

Vd: int a, b;

a = 4;

b = 5;

(a > 5) && (b < 3); // false -> 0

Vd: Trong mệnh đề if sau đây thay vì viết

if (delta == 0) { ... };

Ta có thể viết gọn hơn như sau:

if (!delta) { ... }

Cấu trúc tổng quát của một chương trình C

- Toán tử tăng giảm: vd a++ // a = a + 1

  a-- // a = a -1

- Toán tử gán: a+= b // a = a + b

  a*=b // a = a * b

- Toán tử điều kiện - biểu thức điều kiện

  <dieu-kien> ? <bieu-thuc1> : <bieu-thuc2>;

  ➔ Nếu đk đúng thì thực hiện bt 1, sai thì thực hiên bt 2

  Vd: max = (max > a)? max : a;

Điều kiện và vòng lặp

1. If

Có 3 loại:

- if (bieu_thuc)

lệnh;

vd if (a == b)

printf(“ a va b bang nhau);

- if (bieu_thuc)

lệnh_1;

else

lệnh_2;

vd if (a == b)

printf(“ a va b bang nhau);

else

printf(“a va b khac nhau”);

if (biểu_thức_1)

lenh_1;

else if (biểu_thức_2)

lenh_2;

else if (biểu_thức_3)

lenh_3;

.....

else

lenh_n;

Vd:

```
if(a < 18)
        printf(“suy dinh duong”);
else if(a >= 18 && a < 25)
        printf(“binh thuong”);
else (a >= 25)
        printf(“beo”);
```

switch-case

```
switch (biểu_thức) {
case hằng_1: lệnh_1; break;
 case hằng_2: lệnh_2; break;
…
case hằng_n: lệnh_n; break;
default: lệnh; break;
}
```

Vd:

```
switch (so % 5) {
 case 0: so += 5; printf ("Tri la: %d\n", so); break;
case 1: so += 1; printf ("Tri la: %d\n", so); break;
case 3: so += 3; printf ("Tri la: %d\n", so); break;
default: printf ("Khong thoa\n"); break; }
```

while

- while (bieu-thuc)
  lenh;

vd:

```
int i, a = 0;
while (a <= 10)
    a+=i;
```

- do{
  lenh;
  }
  while (bieu_thuc);

vd:

```
int n;
do{
printf("nhap n = ");
scanf("%d, &n);
}
while(n<0 || n > 100);
```

for

for (<biểu_thức1>; <biểu_thức2>; <biểu_thức3>)

lệnh;

vd:

```
int s = 0;
for (int i = 1; i <= n; i++)
    s += i;
```

Hoặc có thể viết gọn hơn

```
for (int i = 1, s = 0; i <= n; i++)
    s += i;
```

Hàm

Chương trình con là đoạn chương trình đảm nhận thực hiện một thao tác nhất định.

Hàm main() là hàm đặc biệt của C, nó là một hàm mà trong đó các thao tác lệnh (bao gồm các biểu thức tính toán, gọi hàm, ...) được C thực hiện theo một trình tự hợp logic để giải quyết bài toán được đặt ra.

Việc sử dụng hàm sẽ làm cho chương trình trở nên rất dễ quản lý, dễ sửa sai.

Tất cả các hàm đều ngang cấp nhau. Các hàm đều có thể gọi lẫn nhau, dĩ nhiên hàm được gọi phải được khai báo trước hàm gọi

VD:

```
#include<stdio.h>
int so_sanh (int a, int b) {
int ket_qua;
if (a >b) ket_qua = 1;
else if (a == b) ket_qua = 0;
else if (a < b) ket_qua = -1;
 return ket_qua;
}
main() {
int a, b, ket_qua;
 printf ("Moi nhap hai so "); scanf ("%d %d" , &a, &b);
 ket_qua = so_sanh (a, b);
 switch (ket_qua) {
case -1:
```

```
        printf ("So %d nho hon so %d \n" , a, b); break;
case 0: printf ("So %d bang so %d \n", a, b); break;
case 1: printf ("So %d lon hon so %d \n" , a, b); break;
 }
//// khi muốn return kh đc dùng void
void nhap(int a[100], int n){
for(int I = 0; i < n; i ++){
        printf("a[%d] = ", i);
        scanf("%d", &a[i]);
}
}
void xuat(int a[100], int n){
for(int i = 0; I < n; i++)
        printf("%d\t", a[i]);
}
int main(){
int a[100], n;
printf("Nhap n = "); scanf("%d", &n);
nhap(a, n);
printf("\n Cac gia tri trong mang la:\n);
xuat(a, n);
}
```

Mảng

1. Mảng 1 chiều

   Cú pháp khai báo mảng một chiều như sau:

   <kiểu tên_mảng>[<kích_thước>]; Vd: int a[10], b[20] …

   + Với kích_thước là một hằng số nguyên cụ thể, cho biết số phần tử trong chiều đang xét.

   +Trong C, cước số các phần tử của mảng luôn đi từ 0 trở đi, nên mảng một chiều có n phần tử thì cước số các phần tử của mảng là 0,..., n1.

   Vd: int a[7], b[20] …

n	0	1	2	3	4	5	6
a[n]	5	2	3	4	5	6	8

2. Mảng nhiều chiều:

   Cú pháp khai báo mảng nhiều chiều như sau:

   <kiểu tên_mảng>[<kích_thước_chiều1>][<kích_thước_chiều2>] [...];

   Vd: int a[5][5];

	0	**1**	**2**	**3**	**4**
0	5	5		4	65
1	6	3	5	9	2
2	7	2	2	2	3
3	56	2	5	5	5
4	6	3	9	5	2

Vd

```
#include <stdio.h>
```

```
void nhap(int a[100][100], int n, int m)
{
	for(int i = 0; i < n; i++)
		for(int j = 0; j < m; j++)
		{
			printf("a[%d][%d]= ", i,j);
			scanf("%d", &a[i][j]);
		}
}
void xuat(int a[100][100], int n, int m)
{
	for(int i = 0; i < n; i++)
	{
		for(int j = 0; j < m; j++)
			printf("%d\t", x[i][j]);
			printf("\n");
	}
}
int main()
{
	int a[100][100], n, m;
	printf(“nhap so hang va so cot: “); scanf(“%d%d”, &n, &m);
	nhap(a, n, m);
```

```
        printf("\nMa tran:\n);
        xuat(a, n, m);
}
```